1. Assumptions:
2. Financial applications mostly use Big Numbers to overcome fraction errors, but since all the operations are rounded, using floats and integers is assumed accepted.
3. The tax ranges in the table are not month dependent, and the illustration examples did not make any use of the date range in the calculation of the taxes, thus the date range will be just copied from the input to the output as in the examples
4. The percentage column in the input file is assumed to contain an integer to be use as the percentage with no % sign
5. The UI was created by the JDeveloper using drag and drop, so the code was not optimized or cleaned, as it is just a way to test the application
6. Running the application
7. The project was created in JDeveloper 12.2.1.0.0 in case you need to view the code hierarchy.
8. The application requires JRE 8 or newer.
9. The application can be run either using a swing interface or a console interface, both are completely loosely coupled from the main calculation engine, they both monitor the tax calculation progress using the observer design pattern
   1. Running the swing interface
      1. From within JDeveloper, run the class **com.myob.assignment.view.frames.MainFrame**
      2. Outside JDeveloper, double click the jar file below **ViewController/deploy/**
   2. Running the console interface
      1. From within JDeveloper, run the class **com.myob.assignment. model.console.Runner, you will** need to supply arguments through the “Run Configuration” dialog, the first argument is the path to the input CSV file and the second argument is the path to the output CSV file.
      2. From outside JDveloper run java -classpath TaxCalculator.jar com.myob.assignment.model.console.Runner <input\_file\_path> <output\_file\_path>) through the system console, assuming you are in the same directory where you placed the jar file TaxCalculator.jar
10. Testing the application
11. The provided CSV file **input.csv** contains the output columns as well, this is only for easier comparison with the output, they are not used in the data processing.
12. The application includes a few JUnit test cases that can be ran using the maven test goal of the Model project.
13. The CSV file **10000.csv** contains around 10000 entries, they are repeated, it is just intended to be used for performance testing.